# ACL Viewer

*ACL Handling in Java and on the Web*

**Zachary Rice**
Fermilab - Acclerator Division - Controls
Parkland College - CCI

**Contact Information:**
Phone: 217 714 5103
Email: zachrice9@gmail.com

## Abstract

A discourse on the development of an ACL(Accelerator Command Language) application made in Java and Google Web Toolkit. ACL Viewer is an application designed for user friendly ACL accessibility. This discussion covers an introduction to ACL, the functionality of ACL Viewer, the constituents that makeup ACL Viewer, and an overview of the communications between the user and server.

## Introduction

This project was prompted due to the lack of simple cross platform ACL(Accelerator Command Language) script handling applications available at Fermilab.

1. Acl Viewer Application
   - Developed in Java (cross-platform).
   - Scripting and Logging functions.
   - Access to all ACL commands.
2. Acl Viewer Web Application
   - Developed with Google Web Toolkit
   - Only access to `read` command

## ACL

ACL is the Accelerator Command Language authored by Fermilab's Brian Hendricks.

- ACL is a Scripting language.
- Interface with ACNET devices throughout the lab.
- About one hundred commands.
- Easy syntax.
- Ability to define variables and symbols.
- Device oriented commands like: read, set, reset
- Includes control commands like: loop, while, return, wait

Simple example script:

```
wait\sec 5;read G:SCTIME
```

This script tells the OAC to start the following task: wait five second, then read the time in this Super Cycle.

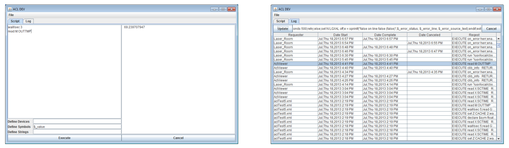

**Figure 1:** Script Tab (left); Log Tab (right)

## GUI

The GUI I developed is clean and simple as seen in Figure 1.

### Features

- User friendly.
- Script tab has two large text fields.
- Log tab has one large table.
- Ability to save/load scripts.
- Ability to save/load logs.
- Area to define symbols
- Mouseover to view ACL scripts in log tab.

## Software/Hardware Interface

The class files that makeup my application are as followed; `AppGUI`, `Reading`, `Setting`, `ACL`, and `Convert`. Simplifying class processes: `AppGUI` creates the GUI and initializes tasks. `ACL` consolidates date from `Reading` and `Setting` then displays it to the GUI. `Reading` and `Setting` sends and receives data. Before the user sends a script down to the OAC, a safe path needs to be established as seen in Figures 2 and 3.

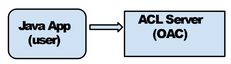


**Figure 2:** User requesting path

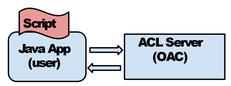


**Figure 3:** Sending path back to user

Script comes into OAC then gets converted to an ACNET message which is sent to CENTRA. CENTRA starts an ACLD tasks and begins to fetch data from devices specified in the script (Figure 4). Once this process is completed it will be sent back up the same path to be displayed by on the GUI as seen in Figure 5.

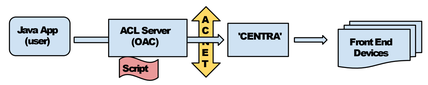


**Figure 4:** Script is being send down to the OAC where it will be converted into an ACNET message which is then sent down to CENTRA. CENTRA will then perform the tasks.

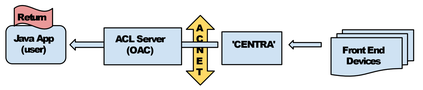


**Figure 5:** Return data is sent back up through the same path

## Scripting

`Reading` and `Setting` have a main function called 'readMe' and 'set' respectively. `ACL` makes two readings by calling 'readMe' twice which returns two arrays of doubles. In addition to the ACL script, the arrays of doubles are used as parameters for `Setting`'s 'set' method in order to tell the OAC what task to execute.

### The scripting process:

1. Make initial reading (slot)
2. Start while loop
3. Make second reading (selection)
4. Make setting with ACL script included
5. Start second while loop
6. Make third reading (data to be displayed after conversion)
7. Convert array of doubles to string
8. Display in GUI

## Logging

As mentioned before the application has a logging functionality to monitor ACL activity as seen in Figure 1(right). Logging requires multiple calls to `Reading`'s `readMe` method which returns an array of doubles. One call to `readMe` will return a chunk of data correlating with the five column headers. The amount of data that can be returned by one call is limited by the OAC so making multiple calls is necessary to fill the entire table. The basic framework to fill the table looks something like:

```
while(no interruptions)
   readings = read(params)
   convert(readings)
   fillTable
```

The program will exit the while loop once the readings return nothing.

## Google Web Toolkit

In addition to the Java application, I developed a web application made with Google Web Toolkit(GWT). Essentially, GWT converts Java code to JavaScript allowing it to be launched on all browsers.

- Similar GUI to Acl Viewer in Java (Figure 6).
- Communicates with a different ACL server (URL access).
- Limited to `read` command.
- No logging abilities.

**Figure 6:** GWT GUI in browser


**Figure 7:** Sending script to URL ACL server


**Figure 8:** Returning values to GUI

User sends script to `http://www-bd.fnal.gov/cgi-bin/acl.pl?acl=CODE]` where 'ACL CODE' is the formatted ACL script. Formatting happens under the hood in the App implementation file.

## References

[1] J. Patrick, *Fermilab Control System("ACNET")*. Batavia, Illinois, Feb 17, 2005.

[2] Andrey Petrov, Fermi National Accelerator Laboratory Accelerator Controls Department *Beyond ACNET: Evolution of Accelerator Control System at Fermilab*. SLAC March 17, 2009.

## Acknowledgements

This work was supported in part by the U.S. Department of Energy (DOE), Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Community College Internships (CCI) Program. I'd like to thank Fermilab, the Accelerator Division, Linden Carmichael, Arden Warner, Brian Hendricks, Glen Johnson, and John DeVoy.